Notice d’utilisation des raccords 1D-3D

Résumé :

Cette notice a pour objectif de guider l'utilisateur dans le choix d'un raccord Poutre – 3D en dynamique dans Code_Aster.

Les méthodologies de raccord proposées sont de deux type :
- Raccord surfacique non-intrusif dénommé bascule 1D-3D (option `3D_POU` de `LIAISON_ELEM`)
- Raccord volumique intrusif dans le cadre Arlequin (option `3D_POU_ARLEQUIN` de `LIAISON_ELEM`)

Table des Matières

1 Introduction

Pour le calcul d'effets localisés en espace et en temps, un modèle poutre permet de gagner en temps de calcul en absence des effets non linéaires, et un modèle poutre-3D mixte permet de prendre en compte les effets non linéaires limités en espace.

2 Bascule non-intrusive 1D-3D

Une bascule d'un modèle poutre à un modèle mixte poutre-3D permet de gagner considérablement en temps de calcul pour une précision équivalente à un modèle 3D entier [bib2].

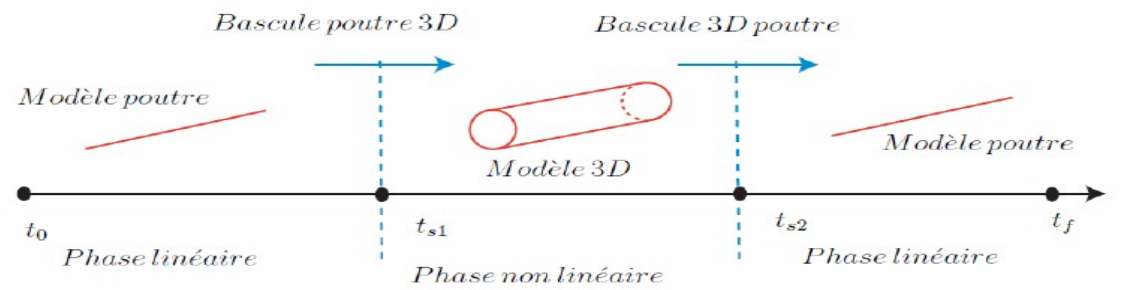


Image 1.1-a fonctions de pondération dans le cadre Arlequin

Code_Aster permet d'initialiser un calcul dynamique avec des champs de déplacements, vitesses et accélérations non nulles (`ETAT_INIT` dans les opérateurs de calcul dynamique). Ces champs d'initialisation peuvent être préalablement calculés, puis transformés en un champ dans *Code_Aster*.

2.1 Principe théorique de la bascule

La bascule consiste à passer d'un modèle de Poutre (**U_p** est la solution poutre du système $M_p \ddot{U}_p + C_p \dot{U}_p + K_p U_p = f_p$) à un modèle 3D (solution $\mathbf{U_{3D}}$ à initialiser) par l'intermédiaire d'une initialisation pertinente de la solution 3D. Le principe de base est le suivant :

1. à partir de la solution poutre $\mathbf{U_p}$, créer une solution 3D $\mathbf{PU_p}$ pour l'hypothèse de section rigide

2. écrire le champs déplacement 3D comme étant la somme du champ extrudé de type Poutre $\mathbf{Pu_p}$ et d'un champ correcteur en déplacement noté $\mathbf{U_{3Dc}}$, prenant en compte la déformation de la section **:**

$$U_{3\mathrm{D}} = PU_p + U_{3\mathrm{Dc}}$$

3. La dynamique du modèle 3D s'écrira alors comme suit :

$$M_{3\mathrm{D}}\left(P\ddot{U}_p + \ddot{U}_{3\mathrm{Dc}}\right) + C_{3\mathrm{D}}\left(P\dot{U}_p + \dot{U}_{3\mathrm{Dc}}\right) + K_{3\mathrm{D}}\left(PU_p + U_{3\mathrm{Dc}}\right) = f_{3\mathrm{D}}$$

4. corriger les déplacements, **en statique**, sous l'effet d'un vecteur force f_{3Dc} calculé comme suit :

$$f_{3\mathrm{Dc}} = K_{3\mathrm{D}} U_{3\mathrm{Dc}} = f_{3\mathrm{Dc}}\left(t = t_b\right) - M_{3\mathrm{D}} P\ddot{U}_p - C_{3\mathrm{D}} P\dot{U}_p - K_{3\mathrm{D}} PU_p$$

La correction statique est effectuée sur 3 pas de temps successifs : l'instant de bascule t_b ainsi que celui d'avant ($t_{b-1} = t_b - \Delta t$) et celui d'après ($t_{b+1} = t_b + \Delta t$) où Δt est le pas de temps de calcul. On en déduit une correction en vitesse qui s'écrit comme suit :

$$\dot{U}_{3D}=\frac{[PU_p+U_{3Dc}]_{(t_{b+1})}-[PU_p+U_{3Dc}]_{(t_{b-1})}}{2\Delta t}$$ (Différences Finies Centrées)

Les accélérations, quant à elles, sont soit calculées de la même façon selon un schéma de type Différences Finies Centrées,

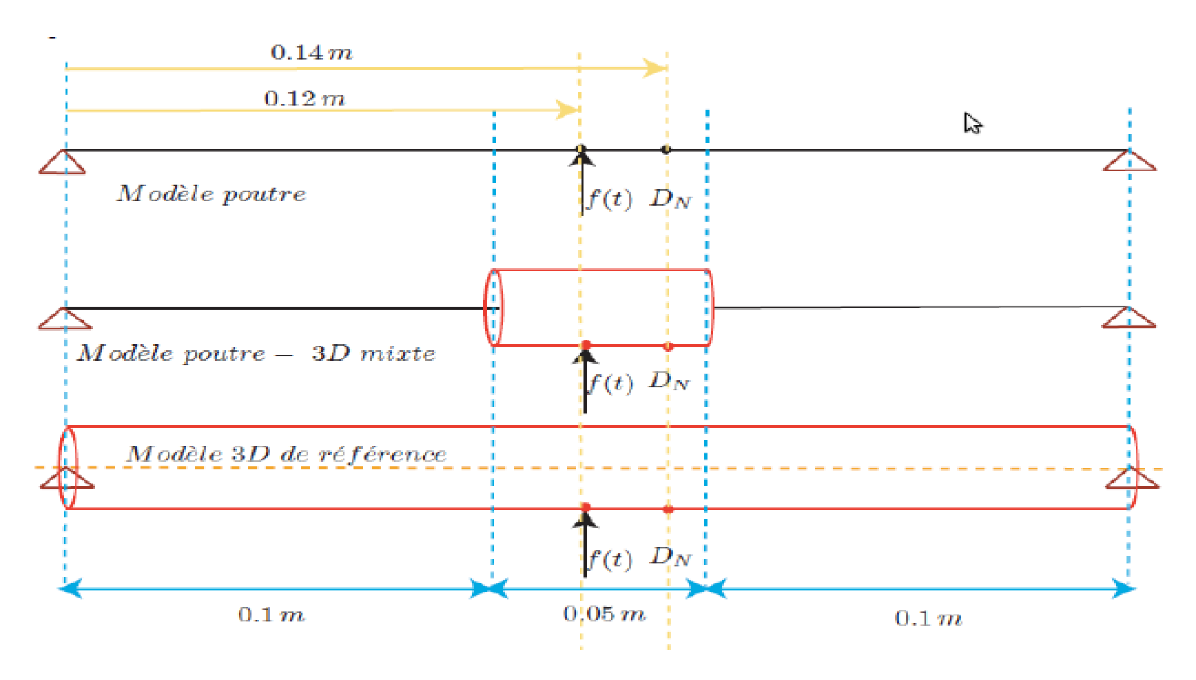
$$\ddot{U}_{3D}=\frac{[PU_p+U_{3Dc}]_{(t_{b-1})}+[PU_p+U_{3Dc}]_{(t_{b+1})}-2[PU_p+U_{3Dc}]_{(t_b)}}{\Delta t^2}$$

soit déterminées de façon automatique par *Code_Aster* dans le cas de l'utilisation d'un schéma d'intégration implicite en temps.

La correction comporte une correction de la flèche ainsi qu'une correction pour prendre en compte la déformation de la section.

2.2 Exemple de mise en œuvre

La méthodologie est explicitée ci-dessous pour un problème de poutre bi-appuyée soumise à une excitation dynamique sinusoïdale durant 3s.



La simulation démarre par un modèle de poutre. Le calcul dynamique sur le mono-modèle de poutre a lieu de l'instant initial t0 jusqu'à l'instant de bascule au modèle 3D (ici, t0=0s, tb=2.4s).

```
Ti=0;
Tf=3.0;
Tb=2.4;

dt=0.0015;

Tb_1=Tb-dt;
Tb_2=Tb+dt;
```

Les opérations suivantes seront utilisées pour la bascule.

```
# Création du maillage Python vide
mail_py = MAIL_PY()
```

```
# Conversion du maillage Aster en maillage Python
mail_py.FromAster('Mail')

# Coordonnées des nœuds
COORD_3D = mail_py.cn

iPouAllNode = mail_py.gno.get('AllNode')

iPouRP1 = mail_py.gno.get('RP1')
iPouRP2 = mail_py.gno.get('RP2')

C3DZ = zeros((len(COORD_3D),1))
for i in range(len(C3DZ)):
   C3DZ[i] = COORD_3D[i][2]

C1DZ1 = []
min=1112
while (min!=1111):
   i = 0
   s = int((abs(C3DZ[i])+ajust)*Pow)/float(Pow)
   for i in range(len(C3DZ)):
      if (int((C3DZ[i]+ajust)*Pow)/float(Pow))==min:
         C3DZ[i] = 1111
      elif( (int((C3DZ[i]+ajust)*Pow)/float(Pow)) < s ):
         s = int((C3DZ[i]+ajust)*Pow)/float(Pow)
   if (min!=s):
      min=s
      C1DZ1.append(min)

C1D = zeros((len(C1DZ1)-1,3))
for i in range(len(C1D)):
   C1D[i][0] = 0
   C1D[i][1] = 0
   C1D[i][2] = C1DZ1[i]

# Fonction pour chercher la section qui correspond a une valeur de z
def OrdreSect(z):
   jk=0
   while(C1D[jk][2]!=z):
      jk=jk+1
    return jk
```

A l'instant de bascule tb, on récupère les déplacements Up, les vitesses Vp et accélérations Ap de la fibre neutre de la poutre par l'opérateur `CREA_CHAMP`, opération `'EXTR'`. Ensuite, les composantes (`DX,DY,DZ,DRX,DRY,DRZ`) sont extraites dans des tables Python.

```
Temps = NP.array([Tb,Tb_1,Tb_2])
nbpas = temps.shape[0]

U0=[]
U1=[]
U2=[]

for i in range(nbpas):

   pas = temps[i]

   DEP=CREA_CHAMP(OPERATION='EXTR',
                  TYPE_CHAM='NOEU_DEPL_R',
```

```
                    RESULTAT=DLT,
                    NOM_CHAM='DEPL',
                    INST=pas,);

   Upx = DEP.EXTR_COMP('DX',['AllNode']).valeurs;
   Upy = DEP.EXTR_COMP('DY',['AllNode']).valeurs;
   Upz = DEP.EXTR_COMP('DZ',['AllNode']).valeurs;
   Uprx = DEP.EXTR_COMP('DRX',['AllNode']).valeurs;
   Upry = DEP.EXTR_COMP('DRY',['AllNode']).valeurs;
   Uprz = DEP.EXTR_COMP('DRZ',['AllNode']).valeurs;

   ACC=CREA_CHAMP(OPERATION='EXTR',
                    TYPE_CHAM='NOEU_DEPL_R',
                    RESULTAT=DLT,
                    NOM_CHAM='ACCE',
                    INST=pas,);

   ddotUpx = ACC.EXTR_COMP('DX',['AllNode']).valeurs;
   ddotUpy = ACC.EXTR_COMP('DY',['AllNode']).valeurs;
   ddotUpz = ACC.EXTR_COMP('DZ',['AllNode']).valeurs;
   ddotUprx = ACC.EXTR_COMP('DRX',['AllNode']).valeurs;
   ddotUpry = ACC.EXTR_COMP('DRY',['AllNode']).valeurs;
   ddotUprz = ACC.EXTR_COMP('DRZ',['AllNode']).valeurs;
```

Par une hypothèse de type corps rigide, on calcule des tables Python contenant les déplacements PUp, les vitesses PVp et accélérations PAp du modèle mixte par extrusion des champs poutre dans l’épaisseur.

Ensuite, des tables Python PUp et ddotPUp sont calculées et contiennent les champs de déplacement et d'accélération des nœuds du modèle mixte en fonction de la topologie (1D ou 3D).

```
   dictLNo = []
  dictPUpx = []
  dictPUpy = []
  dictPUpz = []
  dictPUprx = []
  dictPUpry = []
  dictPUprz = []
  dictAccx = []
  dictAccy = []
  dictAccz = []
  dictAccrx = []
  dictAccry = []
  dictAccrz = []
  for i in range(len(PUp)):
     j=i+1
     if len(PUp[i])==3:
        dictLNo.append('N%i'%j);
        dictPUpx.append(PUp[i][0]);
        dictPUpy.append(PUp[i][1]);
        dictPUpz.append(PUp[i][2]);
        dictAccx.append(ddotPUp[i][0]);
        dictAccy.append(ddotPUp[i][1]);
        dictAccz.append(ddotPUp[i][2]);
     else:
        dictLNo.append('N%i'%j);
        dictPUpx.append(PUp[i][0]);
        dictPUpy.append(PUp[i][1]);
        dictPUpz.append(PUp[i][2]);
        dictPUprx.append(PUp[i][3]);
```

```
        dictPUpry.append(PUp[i][4]);
        dictPUprz.append(PUp[i][5]);
        dictAccx.append(ddotPUp[i][0]);
        dictAccy.append(ddotPUp[i][1]);
        dictAccz.append(ddotPUp[i][2]);
        dictAccrx.append(ddotPUp[i][3]);
        dictAccry.append(ddotPUp[i][4]);
        dictAccrz.append(ddotPUp[i][5]);
```

Ces tables Python sont par la suite transformées en listes puis en champs de Code_Aster.

```
TD=CREA_TABLE(LISTE=(_F(LISTE_K=dictLNo,PARA='NOEUD'),
                     _F(LISTE_R=dictPUpx,PARA='DX'),
                     _F(LISTE_R=dictPUpy,PARA='DY'),
                     _F(LISTE_R=dictPUpz,PARA='DZ'),
                     _F(LISTE_R=dictPUprx,PARA='DRX'),
                     _F(LISTE_R=dictPUpry,PARA='DRY'),
                     _F(LISTE_R=dictPUprz,PARA='DRZ'),))

TA=CREA_TABLE(LISTE=(_F(LISTE_K=dictLNo,PARA='NOEUD'),
                     _F(LISTE_R=dictAccx,PARA='DX'),
                     _F(LISTE_R=dictAccy,PARA='DY'),
                     _F(LISTE_R=dictAccz,PARA='DZ'),
                     _F(LISTE_R=dictAccrx,PARA='DRX'),
                     _F(LISTE_R=dictAccry,PARA='DRY'),
                     _F(LISTE_R=dictAccrz,PARA='DRZ'),))

DeplPUp=CREA_CHAMP(TYPE_CHAM='NOEU_DEPL_R',
                   OPERATION='EXTR',
                   MAILLAGE=Mail,
                   TABLE=TD,);

AccIni=CREA_CHAMP(TYPE_CHAM='NOEU_DEPL_R',
                  OPERATION='EXTR',
                  MAILLAGE=Mail,
                  TABLE=TA,);
```

Les produits matrices-champs **K3D*PUp** et **M3D*PAp** sont effectués par `PROD_MATR_CHAM`.

```
FORCE1=PROD_MATR_CHAM(MATR_ASSE=RIGIDITE,
                      CHAM_NO=DeplPUp,
                      TITRE='PROD_MATR_CHAM1',);

TBLf1=POST_RELEVE_T(ACTION=_F(OPERATION='EXTRACTION',
                    INTITULE='f1',
                    CHAM_GD=FORCE1,
                    GROUP_NO='AllNode',
                    NOM_CMP=('DX','DY','DZ','DRX','DRY','DRZ',),),);

VARf1=TBLf1.EXTR_TABLE()

FORCE2=PROD_MATR_CHAM(MATR_ASSE=MASSE,
                      CHAM_NO=AccIni,
                      TITRE='PROD_MATR_CHAM2',);

TBLf2=POST_RELEVE_T(ACTION=_F(OPERATION='EXTRACTION',
                    INTITULE='f2',
                    CHAM_GD=FORCE2,
                    GROUP_NO='AllNode',
                    NOM_CMP=('DX','DY','DZ','DRX','DRY','DRZ',),),);
```

```
   VARf2=TBLf2.EXTR_TABLE()
```

En fonction des forces en présence dans le problème traité, le chargement **f3D(tb)-K3D*PUp-M3D*PAp** est calculé sous forme d'un dictionnaire par nœuds puis puis stocké comme suit :

```
CHARGE3D=AFFE_CHAR_MECA(MODELE=MODELE,
                        FORCE_NODALE=(dictCharge),);
```

Ensuite, un champ de correction de la déformation de la section U3Dc est calculé par `MECA_STATIQUE` comme la réponse, à l'instant de bascule, du modèle 3D de référence au chargement **f3D(tb)-K3D*PUp-M3D*PAp**.

```
 Mstat=MECA_STATIQUE(MODELE=MODELE,
                     CHAM_MATER=CHMAT,
                     CARA_ELEM=Carel3D,
                     EXCIT=(_F(CHARGE=CondLim,),
                            _F(CHARGE=CHARGE3D,),),);
```

La réponse qui en résulte est relevée puis stockée dans une table :

```
tableUc=POST_RELEVE_T(ACTION=_F(OPERATION='EXTRACTION',
                      INTITULE='U3Dc',
                      RESULTAT=Mstat,
                      NOM_CHAM='DEPL',
                      PRECISION=1e-15,
                      GROUP_NO='AllNode',
                      NOM_CMP=('DX','DY','DZ', 'DRX', 'DRY', 'DRZ',),),);

TC=tableUc.EXTR_TABLE()
```

Enfin, le champ de déplacement du patch 3D du modèle mixte à l'instant **tb** sera la somme des deux champs **U3Dc** et **PUp** .

```
# dictionnaire des déplacements au moment de la bascule
dictDeplx = []
dictDeply = []
dictDeplz = []
dictDeplrx = []
dictDeplry = []
dictDeplrz = []
for i in range(len(U0)):
   j=i+1
   if len(U0[i])==3:
      dictDeplx.append(U0[i][0]);
      dictDeply.append(U0[i][1]);
      dictDeplz.append(U0[i][2]);
   else:
      dictDeplx.append(U0[i][0]);
      dictDeply.append(U0[i][1]);
      dictDeplz.append(U0[i][2]);
      dictDeplrx.append(U0[i][3]);
      dictDeplry.append(U0[i][4]);
      dictDeplrz.append(U0[i][5]);

Dini=CREA_TABLE(LISTE=(_F(LISTE_K=dictLNo,PARA='NOEUD'),
                       _F(LISTE_R=dictDeplx,PARA='DX'),
                       _F(LISTE_R=dictDeply,PARA='DY'),
                       _F(LISTE_R=dictDeplz,PARA='DZ'),
                       _F(LISTE_R=dictDeplrx,PARA='DRX'),
```

```
                       _F(LISTE_R=dictDeplry,PARA='DRY'),
                       _F(LISTE_R=dictDeplrz,PARA='DRZ'),))

Dep0=CREA_CHAMP(TYPE_CHAM='NOEU_DEPL_R',
                OPERATION='EXTR',
                MAILLAGE=Mail,
                TABLE=Dini,);
```

Les champs de vitesse et accélération sont également calculées selon un schéma de Différences Centrées, en utilisant les champs Poutre et corrigés des déplacements, vitesses et accélérations à l'instant de bascule **tb** et aux instants **tb – Δt** et **tb + Δt**.

```
# Calcul des vitesses corrigées
dictVitex = []
dictVitey = []
dictVitez = []
dictViterx = []
dictVitery = []
dictViterz = []
for i in range(len(U1)):
    j=i+1
    if len(U1[i])==3:
       dictVitex.append((U2[i][0]-U1[i][0])/(2*dt));
       dictVitey.append((U2[i][1]-U1[i][1])/(2*dt));
       dictVitez.append((U2[i][2]-U1[i][2])/(2*dt));
    else:
       dictVitex.append((U2[i][0]-U1[i][0])/(2*dt));
       dictVitey.append((U2[i][1]-U1[i][1])/(2*dt));
       dictVitez.append((U2[i][2]-U1[i][2])/(2*dt));
       dictViterx.append((U2[i][3]-U1[i][3])/(2*dt));
       dictVitery.append((U2[i][4]-U1[i][4])/(2*dt));
       dictViterz.append((U2[i][5]-U1[i][5])/(2*dt));

Vini=CREA_TABLE(LISTE=(_F(LISTE_K=dictLNo,PARA='NOEUD'),
                       _F(LISTE_R=dictVitex,PARA='DX'),
                       _F(LISTE_R=dictVitey,PARA='DY'),
                       _F(LISTE_R=dictVitez,PARA='DZ'),
                       _F(LISTE_R=dictViterx,PARA='DRX'),
                       _F(LISTE_R=dictVitery,PARA='DRY'),
                       _F(LISTE_R=dictViterz,PARA='DRZ'),))

Vit0=CREA_CHAMP(TYPE_CHAM='NOEU_DEPL_R',
                OPERATION='EXTR',
                MAILLAGE=Mail,
                TABLE=Vini,);

# Calcul des accélérations corrigées
dictAccex = []
dictAccey = []
dictAccez = []
dictAccerx = []
dictAccery = []
dictAccerz = []
for i in range(len(U1)):
    j=i+1
    if len(U1[i])==3:
       dictAccex.append((U2[i][0]-2*U0[i][0]+U1[i][0])/(dt**2));
       dictAccey.append((U2[i][1]-2*U0[i][1]+U1[i][1])/(dt**2));
       dictAccez.append((U2[i][2]-2*U0[i][2]+U1[i][2])/(dt**2));
    else:
```

```
        dictAccex.append((U2[i][0]-2*U0[i][0]+U1[i][0])/(dt**2));
        dictAccey.append((U2[i][1]-2*U0[i][1]+U1[i][1])/(dt**2));
        dictAccez.append((U2[i][2]-2*U0[i][2]+U1[i][2])/(dt**2));
        dictAccerx.append((U2[i][3]-2*U0[i][3]+U1[i][3])/(dt**2));
        dictAccery.append((U2[i][4]-2*U0[i][4]+U1[i][4])/(dt**2));
        dictAccerz.append((U2[i][5]-2*U0[i][5]+U1[i][5])/(dt**2));

Aini=CREA_TABLE(LISTE=(_F(LISTE_K=dictLNo,PARA='NOEUD'),
                       _F(LISTE_R=dictAccex,PARA='DX'),
                       _F(LISTE_R=dictAccey,PARA='DY'),
                       _F(LISTE_R=dictAccez,PARA='DZ'),
                       _F(LISTE_R=dictAccerx,PARA='DRX'),
                       _F(LISTE_R=dictAccery,PARA='DRY'),
                       _F(LISTE_R=dictAccerz,PARA='DRZ'),))

Acc0=CREA_CHAMP(TYPE_CHAM='NOEU_DEPL_R',
                OPERATION='EXTR',
                MAILLAGE=Mail,
                TABLE=Aini);
```

Les champs d'initialisation ainsi construits sont utilisés pour initialiser le calcul dynamique (`ETAT_INIT` dans les opérateurs de calcul dynamique).

```
Bascule=DYNA_LINE_TRAN(MODELE=MODELE,
                       CHAM_MATER=CHMAT,
                       MATR_MASS=MASSE,
                       MATR_RIGI=RIGIDITE,
                       SCHEMA_TEMPS=_F(...),
                       ETAT_INIT=_F(DEPL=Dep0,
                                    VITE=Vit0,
                                    ACCE=Acc0,),
                       EXCIT=(_F(VECT_ASSE=F_Xass,
                                 FONC_MULT=FSIN),),
                       INCREMENT=_F(LIST_INST=LIST,),);
```

La procédure non intrusive de bascule d'un modèle de poutre à un modèle 3D en dynamique transitoire, développée et validée dans Code_Aster (cas-test SDNV139), permet de gagner en temps de calcul car le modèle global de type structure mince peut remplacer le modèle local 3D en l'absence de non linéarités localisées en temps. Par ailleurs, l'utilisation du raccord 3D-POU préexistant dans Code_Aster permet de réduire la taille du modèle pour des cas de non linéarités localisées en espace.

3 Raccord 3D-Poutre dans le cadre Arlequin

Le cadre Arlequin (option `3D_POU_ARLEQUIN` du mot clé `LIAISON_ELEM` de l'opérateur `AFFE_CHAR_MECA`) est une méthode flexible de raccord de modèles 1D et 3D avec recouvrement [bib2]. Elle permet donc également de gagner considérablement en temps de calcul pour une précision équivalente à un modèle 3D entier.

3.1 Principaux ingrédients Arlequin

La méthode Arlequin repose sur les principes suivants :

- Le raccord des sous domaines par l'intermédiaire d'une formulation faible : l'introduction des multiplicateurs de Lagrange dans la zone de collage garantit le couplage des modèles, la continuité des quantités cinématiques, ainsi que le contrôle des écarts des contraintes et des déformations entre les zones couplées;
- La distribution de l'énergie entre domaines et modèles : dans le but de ne pas compter deux fois l'énergie du système global dans la zone de recouvrement, les travaux virtuels associés

aux deux modèles sont distribués entre les sous-domaines couplés à travers la zone de collage par le biais de fonctions de pondération $(\gamma, 1-\gamma)$ qui forment une partition de l'unité (la somme des deux fonctions est égale à 1) sur l'ensemble du domaine d'étude.

La pondération permet d'éviter de prendre en compte l'énergie plusieurs fois dans la même zone et autorise ainsi une certaine liberté à l'utilisateur pour le choix du modèle prédominant. En effet, elle permet de mettre le poids sur le modèle que l'on souhaite faire exprimer.

Les matrices de couplage obtenues sont transformées en relations cinématiques. Ce raccord se traduit donc par des relations linéaires reliant les déplacements de l'ensemble de nœuds 3D (3 degrés de liberté par nœud) liés avec l'ensemble des nœuds de poutre (6 degrés de liberté par nœud).

3.2 Exemple de mise en œuvre

On rappelle que le raccord 3D-Poutre Arlequin, tel que développé aujourd'hui dans Code_Aster, doit satisfaire une exigence très importante. En effet, les maillages 1D et 3D doivent être « hiérarchiquement compatibles », au sens où tous les éléments 3D sont inclus dans l'espace cylindrique de l'élément 1D en vis-à-vis (pas d'éléments 3D à cheval entre deux éléments poutre).

Le maillage associé au modèle mixte 1D-3D est lu par `LIRE_MAILLAGE` :

```
MAIL=LIRE_MAILLAGE(UNITE=20,);
```

La lecture des maillages 1D et 3D peut aussi se faire séparément. Ensuite, les deux maillages sont assemblés via l'opérateur `ASSE_MAILLAGE`.

Ensuite, les éléments de structure et volumiques sont affectés au modèle :

```
MODELE=AFFE_MODELE(MAILLAGE=MAIL,
                          AFFE=(_F(GROUP_MA=('Ref3D'),
                                   PHENOMENE='MECANIQUE',
                                   MODELISATION='3D',),
                                _F(GROUP_MA=('Poutre'),
                                   PHENOMENE='MECANIQUE',
                                   MODELISATION='POU_D_T',),),);
```

Les caractéristiques géométriques des éléments de poutre sont également affectées :

```
CAREL3D=AFFE_CARA_ELEM(MODELE=MODELE,
                             POUTRE=_F(GROUP_MA=('Poutre'),
                                       SECTION='CERCLE',
                                       CARA='R',
                                       VALE=0.005,),);
```

La gestion des pondérations dans les zones libre et de collage Arlequin est laissée aux soins de l'utilisateur. Comme il s'agit d'un couplage de type L2, il suffit d'imposer les coefficients de pondération dans la définition des paramètres matériaux (masse volumique `RHO` et module de Young `E`) relatifs aux différentes zones du recouvrement.

```
# Matériau utilisé dans les zones hors du recouvrement
       MATL=DEFI_MATERIAU(ELAS=_F(E=2.E11*1.0,
                                  NU=0.3,
                                  RHO=7800.0*1.0,),);

          # Matériau utilisé dans la zone de collage incluse dans le
recouvrement
       MATC=DEFI_MATERIAU(ELAS=_F(E=2.E11*0.5,
                                  NU=0.3,
                                  RHO=7800.0*0.5,),);
```

```
# Matériau utilisé dans la zone libre 3D incluse dans le recouvrement
MATR3D=DEFI_MATERIAU(ELAS=_F(E=2.E11*0.99,
                             NU=0.3,
                             RHO=7800.0*0.99,),);

# Matériau utilisé dans la zone libre 1D incluse dans le recouvrement
MATR1D=DEFI_MATERIAU(ELAS=_F(E=2.E11*0.01,
                             NU=0.3,
                             RHO=7800.0*0.01,),);
```

En dehors de la zone de recouvrement, les matériaux sont donnés par l'utilisateur comme il l'entend (matériau `MATL` pour le `GROUP_MA='Poutre'`). Pour la zone de recouvrement, la partition de l'unité est assurée en affectant aux groupes de mailles pré-définies les matériaux déjà pondérés. Dans le cas présent, les modélisations sont pondérées (poids = 50%) de la même façon dans la zone de collage (matériau `MATC` pour les `GROUP_MA=('1DCol','3DCol')`). Dans la zone libre (au sens libre des conditions de raccord Arlequin), nous avons choisi de mettre le poids à 99% sur le modèle volumique (matériau `MATR3D` pour le `GROUP_MA='3DLib'`) et à 1% sur le modèle Poutre (matériau `MATR1D` pour le `GROUP_MA='1DLib'`).

```
CHMAT=AFFE_MATERIAU(MAILLAGE=MAIL,
               AFFE=(_F(GROUP_MA='Poutre',
                        MATER=MATL,),
                     _F(GROUP_MA='1DLib',
                        MATER=MATR1D,),
                     _F(GROUP_MA='1DCol',
                        MATER=MATC,),
                     _F(GROUP_MA='3DLib',
                        MATER=MATR3D,),
                     _F(GROUP_MA='3DCol',
                        MATER=MATC,),
                      ),);
```

La condition de raccord Arlequin est ensuite spécifiée sous l'option `3D_POU_ARLEQUIN` du mot clé `LIAISON_ELEM` de l'opérateur `AFFE_CHAR_MECA`. Pour chaque occurrence du raccord Arlequin, l'utilisateur doit définir les opérandes suivantes :

- Le groupe de mailles 3D de la zone de collage (mot clé `GROUP_MA_1`)
- Le groupe de mailles 1D de la zone de collage (mot clé `GROUP_MA_2`)
- Le concept `CHAM_MATER` définissant les matériaux (servant à assurer la partition de l'unité)
- Le concept `CARA_ELEM` définissant les caractéristiques géométriques servant au calcul des matrices de couplage

```
ARLE=AFFE_CHAR_MECA(MODELE=MODELE,
                    LIAISON_ELEM=(_F(OPTION='3D_POU_ARLEQUIN',
                                     GROUP_MA_1='3DCol',
                                     GROUP_MA_2='1DCol',
                                     CHAM_MATER=CHMAT,
                                     CARA_ELEM=CAREL3D,),
                                  ),);
```

En vue d'un calcul modal ou dynamique sur base modale, les matrices de masse et raideur peuvent être assemblées avec, entre autres, la charge issue du couplage dans le cadre Arlequin.

```
ASSEMBLAGE(MODELE=MODELE,
           CHAM_MATER=CHMAT,
           CARA_ELEM=CAREL3D,
           CHARGE=(CondLim,ARLE),
           NUME_DDL=CO('NUMEDDL'),
           MATR_ASSE=(_F(MATRICE=CO('RIGIDITE'),
                         OPTION='RIGI_MECA',),
```

```
                                             _F(MATRICE=CO('MASSE'),
                                                OPTION='MASS_MECA',),
                                             ),);
```

Un exemple de calcul en dynamique linéaire transitoire en présence sur la base d'un modèle mixte 1D-3D raccordé dans le cadre Arlequin est donné ci-dessous (cf. cas-tests SSLV156 et SSLV160) :

```
RefM3D=DYNA_LINE_TRAN(MODELE=MODELE,
                      CHAM_MATER=CHMAT,
                      CARA_ELEM=CAREL3D,
                      MATR_MASS=MASSE,
                      MATR_RIGI=RIGIDITE,
                      SCHEMA_TEMPS=_F(SCHEMA='NEWMARK',
                                      GAMMA=0.5+alpha,
                                      BETA=(1+alpha)**2/4,),
                      EXCIT=(_F(CHARGE=CondLim,),
                             _F(CHARGE=ARLE,),
                             _F(CHARGE=Charge,
                                FONC_MULT=FSIN),),
                      INCREMENT=_F(LIST_INST=LIST3D,),);
```

4 Bibliographie

[1] M.Tannous, « Développement et évaluation d'approches de modélisation numérique couplées 1D et 3D du contact rotor-stator », Thèse de l'École Centrale Nantes.

[2] A. Ghanem, « Contribution à la modélisation avancée des machines tournantes en dynamique transitoire dans le cadre Arlequin », Thèse de l'INSA de Lyon.